Fuzzy-logic basics: intuitive rules replace complex math

David I Brubaker, The Huntington Group

Although "fuzzy logic" may seem to imply imprecision, it's based on a reliable and rigorous discipline. Fuzzy logic lets you accurately describe control systems in words instead of complicated math.

Fuzzy logic, based on fuzzy set theory, allows you to express the operational and control laws of a system linguistically—in words. Although such an approach might seem inadequate, it can actually be superior to (and much easier than) a more mathematical approach. The main strength of fuzzy set theory, a generalization of classical set theory, is that it excels in dealing with imprecision.

In classical set theory, an item is either a part of a set or not. There is no in-between; there are no partial members. For example, a cat is a member of the set of mammals, and a frog is not. Such sets are called crisp sets.

Fuzzy set theory recognizes that very few crisp sets actually exist. The crisp set of mammals, for example, encounters a problem with the platypus. We have to consider the platypus a full member of the set, even though it has a duck-like bill and lays eggs. Fuzzy set theory doesn't have to deal with such exceptions (or with a number of paradoxes that arise from the strict member/nonmember dichotomy).

Fuzzy logic allows partial set membership; it allows gradual transitions between being fully a member of the set and fully not a member of the set. Being partially a member of a given set, a given element is also partially not a member of that set.

Because fuzzy set theory allows both full membership and full nonmembership (although not simultaneously), it is a generalization of classical set theory. You can actually use fuzzy logic to implement crisp systems, although there is little reason to. Most of the action in fuzzy systems occurs in the transitions—the partial-membership regions of a set.

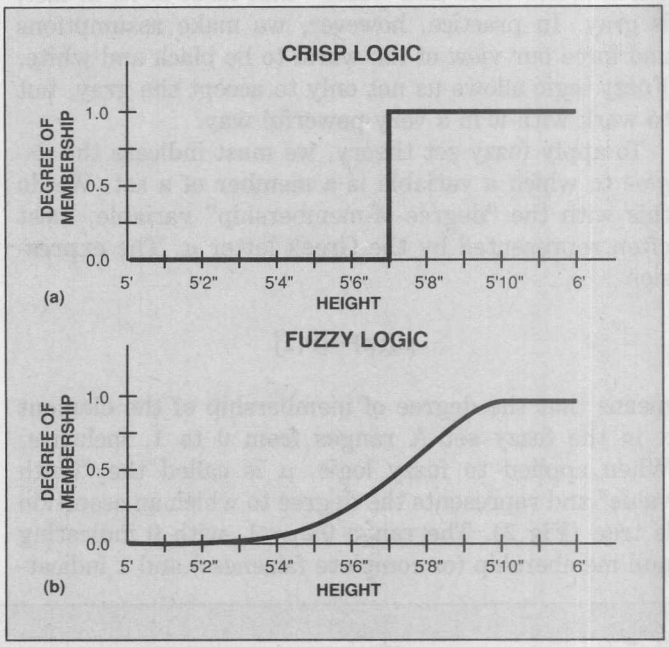


Fig 1—In traditional logic (a), a woman is either a member of the set TALL WOMEN or not, and a mere fraction of an inch can make the difference. Fuzzy logic (b) allows partial set membership that is characterized by a gradual transition from not tall to tall.

Traditional logic recognizes only full or null membership in a set and requires that a given assertion be either true or false. Fuzzy logic, however, allows partial truth and partial falseness. **Fig 1** illustrates the difference.

In classical set theory, we may ask whether Mary, a woman who is 5 ft, 8 in. tall, belongs to the set of tall women. In logic, we ask whether the statement "Mary is a tall woman" is true or false; using bilevel logic, we must select one or the other. If we say that Mary is tall, would she still be tall if she were a quarter inch shorter? Or half an inch shorter?

Fuzzy logic allows the statement "Mary is tall" to have a range of truthfulness, depending on Mary's height. If Mary is 5 ft even, the assertion that she is tall is completely false, but if she is 6 ft even, the assertion is completely true. If she is 5 ft, 8 in. tall, the assertion may be 75% true.

Working with this premise—that the path from truth to falseness may be gradual (and also, implicitly, that something can be simultaneously partially true and partially false)—requires a new mindset, especially for Western engineers. In principle, we agree that the world is not black and white—that most of it, in fact, is gray. In practice, however, we make assumptions and force our view of the world to be black and white. Fuzzy logic allows us not only to accept the gray, but to work with it in a very powerful way.

To apply fuzzy set theory, we must indicate the *degree* to which a variable is a member of a set. We do this with the "degree-of-membership" variable, most often represented by the Greek letter μ. The expression

$$\mu_A(x) \rightarrow [0,1]$$

means that the degree of membership of the element x in the fuzzy set A ranges from 0 to 1, inclusive. When applied to fuzzy logic, μ is called the "truth value" and represents the degree to which an assertion is true (**Fig 2**). The range $0 \leq \mu \leq 1$, with 0 indicating null membership (or complete falseness) and 1 indicating full membership (or complete truth) is consistent with notation used in traditional bilevel logic.

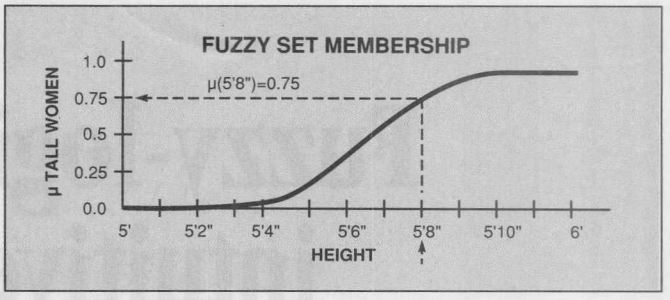


Fig 2—The degree to which an element is a member of a fuzzy set, denoted μ, can range from 0 (null membership) to 1 (full membership). In this example, a woman 5 ft, 8 in. tall has a degree of membership in the set TALL WOMEN of 0.75.

Just as bilevel logic has logical operators for combining logic variables, so does fuzzy logic. Variables in the two logic systems necessarily have different definitions, but they use the same operators: AND, OR, and NOT. The definitions most commonly used are those proposed by Lotfi Zadeh, the creator of fuzzy logic (see **box**, "Fuzzy perspective").

The AND of two fuzzy-logic variables, by Zadeh's definition, is the minimum truth value. That is, for fuzzy variables A and B,

$$\mu_{(A \text{ AND } B)} = \min(\mu_A, \mu_B).$$

The OR of two fuzzy-logic variables is the maximum truth value. Again, for fuzzy variables A and B,

$$\mu_{(A \text{ OR } B)} = \max(\mu_A, \mu_B).$$

The NOT of a fuzzy logic variable is given by

$$\mu_{(\text{NOT } A)} = 1 - \mu_A.$$

Fuzzy perspective

Fuzzy set theory was "created" by Lotfi Zadeh in 1965 and is based on work in multivalued logic by a number of mathematicians earlier in the century. Zadeh was, and is, a professor of electrical engineering and computer science at the University of California, Berkeley.

In his work with complex, nonlinear systems, Zadeh observed that the more complex a system becomes, the less meaningful are low-level details in describing overall system operation. The drive for precision in complex-system description actually becomes a stumbling block; precision often is not only unnecessary, but counterproductive. One of Zadeh's basic assumptions is that a system's operational and control laws can be expressed linguistically—in words.

Fuzzy logic is itself not "fuzzy," but a rigorous mathematical discipline. Neither is it merely another way of looking at probability theory, although both disciplines deal with uncertainty. Probability theory deals with the uncertainty that results from random behavior, whereas fuzzy theory addresses the uncertainty resulting from boundary conditions. The two disciplines are complementary in that there can be (and are) fuzzy stochastic systems, but one does not replace or eliminate the need for the other.

All three of these operators are equivalent to their respective counterparts in bilevel logic for μ limited to 0 and 1.

Plain words replace complex math

In order to work more easily with systems that are too complex to model accurately with mathematics, fuzzy logic resorts to linguistic variables. It is very difficult to express mathematically even the basic control laws involved in driving a car, say, but a verbal description of how to drive—that is, how to react to the various situations that are presented to the driver—is actually quite simple. Any such description, however, must necessarily use imprecise terms such as fast, slow, hard, and soft. (The latter two, for example, describe how much to apply either the accelerator or the brake.)

Fuzzy-set theory accommodates such imprecise terms. For example, the linguistic variable SPEED

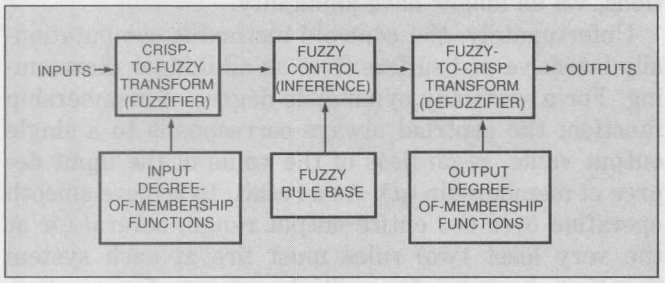


Fig 3—A fuzzy rule-based system has three major components. First, a crisp-to-fuzzy transform changes crisp inputs into degrees of membership, or truth values. Next, an inference mechanism determines actions to take by applying the truth values as conditions to rules. Finally, a fuzzy-to-crisp transform converts fuzzy actions into crisp actions and combines them to form a single, executable action.

might have as values (among others) the fuzzy sets VERY SLOW, SLOW, MEDIUM FAST, FAST, and VERY FAST. It will also have as a value the crisp set STOPPED.

A degree-of-membership function represents each of the fuzzy sets and acts as a transform between the crisp real world and our fuzzy view of the real world. For example, a SPEED of 70 mph may have degrees of membership in each of the fuzzy sets:

$$\mu_{\text{VERY SLOW}}(70\text{ mph}) = 0$$
$$\mu_{\text{SLOW}}(70\text{ mph}) = 0$$
$$\mu_{\text{MEDIUM FAST}}(70\text{ mph}) = 0.3$$
$$\mu_{\text{FAST}}(70\text{ mph}) = 0.8$$
$$\mu_{\text{VERY FAST}}(70\text{ mph}) = 0.4.$$

In fuzzy-logic control algorithms, degrees of membership serve as inputs. The determination of appropriate degree-of-membership functions is part of the design process.

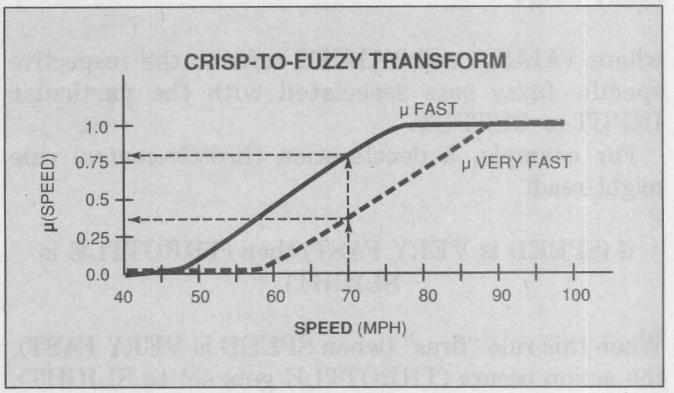


Fig 4—Degree-of-membership functions transform crisp inputs into fuzzy degrees of membership. Here, a speed of 70 mph has membership in the fuzzy set FAST with $\mu = 0.80$ and in the set VERY FAST with $\mu = 0.38$.

The rule-based system is currently the most popular application of fuzzy logic. Its basic structure (**Fig 3**) has three major sections: a crisp-to-fuzzy transform ("fuzzifier"), an inference mechanism that employs rules, and a fuzzy-to-crisp transform ("defuzzifier").

In using such a system, we transform into a fuzzy domain, manipulate the data, and transform back into the crisp domain. This approach is analogous to working in the frequency domain on transformed time-domain data. Because the necessary processing is easier in the frequency domain than it is in the time domain, processing time-domain data warrants the expense of an FFT and an inverse FFT. In a fuzzy system, the base of rules describing system operation *in fuzzy terms* is easy to work with. Consequently, we transform crisp inputs and outputs into a fuzzy domain of intuitive, linguistic rules rather than transforming fuzzy rules into the crisp domain.

As indicated in **Fig 4**, the crisp-to-fuzzy transform is a mapping of an actual crisp value to a degree of membership via an input degree-of-membership function. The resulting degree of membership then becomes an input to the next system block, the inference mechanism.

In the inference mechanism, inputs and truth values serve as conditions for the rules that make up the rule base. At regular intervals, the fuzzy controller samples inputs and applies them to the rule base, resulting in appropriate system outputs. Theoretically, the rule base should cover all possible combinations of input values, but such coverage is typically neither practical nor necessary.

The general form of each rule is:

if (INPUT is VALUE1) then (ACTION is VALUE2),

where VALUE1 and VALUE2 refer to the respective specific fuzzy sets associated with the particular INPUT or OUTPUT.

For example, a deceleration throttle-control rule might read:

if (SPEED is VERY FAST) then (THROTTLE is SLIGHT).

When this rule "fires" (when SPEED is VERY FAST), the action occurs (THROTTLE gets set to SLIGHT). In contrast with other rule-based systems, the action occurs only to the degree that the input is true. If SPEED is VERY FAST with $\mu = 0.25$, then THROTTLE gets set to SLIGHT also with $\mu = 0.25$. This partial setting to SLIGHT occurs in a step that performs combination and defuzzification.

Four ways to defuzzify

A method of combining actions is necessary because more than one rule may fire for any given set of inputs. In addition, the resulting single action (combined from the actions of triggered rules) must be transformed from a fuzzy value to a crisp, executable value. There are currently four popular combination/defuzzification techniques.

The maximizer technique takes the maximum degree-of-membership value from the various triggered rules and performs the corresponding single action. If, for example, as a result of three rules having fired, the THROTTLE mentioned above has

$$\mu_{SLIGHT} = 0.75,$$
$$\mu_{SLIGHT} = 0.40,$$
$$\mu_{MEDIUM} = 0.20,$$

then the throttle setting associated with $\mu_{SLIGHT} = 0.75$ will result, because 0.75 is greater than the other two values of μ. If two actions have the same μ value, and that value is the maximum μ, then some form of conflict resolution is necessary. One possibility is to use an average of the corresponding outputs; another is to select the action associated with a rule's position in the rule base.

The maximizer technique is the simplest combination/defuzzification approach, but it ignores potentially important actions. Another technique, the weighted-average method, averages the various actions after assigning weights based on degree-of-membership values. Although conceptually strong and not computationally demanding, this approach suffers from ambiguity in the output function, as does the maximizer method.

Ambiguity arises because an output degree-of-membership function can specify more than one output value for a given value of μ. An output membership function typically is shaped like a pyramid or a truncated pyramid. If $\mu = 0.5$, the output value can come from the function's rising or falling edge. Worse yet, for a truncated pyramid, $\mu = 1.0$ corresponds to a whole range of values.

It is possible to eliminate ambiguity by mapping output-function components through the specific rules back to input functions. The procedure is tedious, however, and disallows the use of negated input functions as rule conditions. (For example, we could not say "if (SPEED is NOT FAST)".)

The centroid method, illustrated in **Fig 5**, results in an output action that is associated with the center of mass of the active rule outputs. Because we are no longer using the edges of degree-of-membership functions, we no longer have ambiguity.

Unfortunately, the centroid method is computationally intensive and suffers from an additional shortcoming. For a vertically symmetric degree-of-membership function, the centroid always corresponds to a single output value, regardless of the value of the input degree of membership (μ). As a result, to achieve smooth operation over the entire output range, several (or at the very least two) rules must fire at each system iteration. In order for multiple rules to fire at once, input degree-of-membership functions must overlap. Despite these shortcomings, the centroid method is currently the best technique for combination and de-

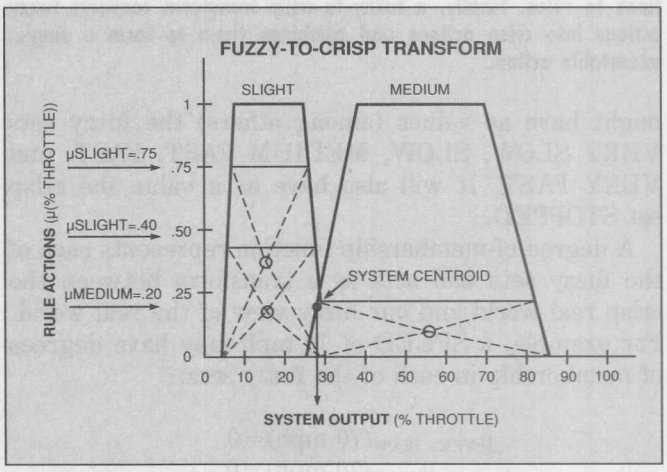


Fig 5—The centroid method combines several fuzzy output actions to form a single executable action. This single output is a weighted average of the centroids of each active degree-of-membership function. This example shows three active rule actions, specifying SLIGHT throttle twice ($\mu = 0.75$ and $\mu = 0.40$) and MEDIUM throttle once ($\mu = 0.20$). Function centroids are marked with small circles. The resulting output is a throttle setting of 29% full throttle.

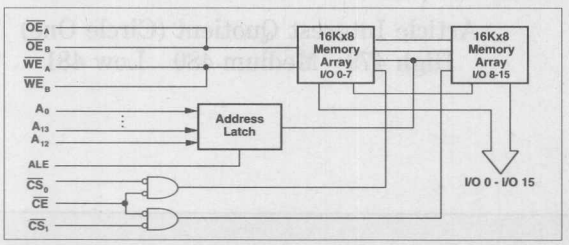
Fast Fast Fast SRAMs for 50MHz Applications

CMOS Cache RAMs for use with X86 & RISC Architectures

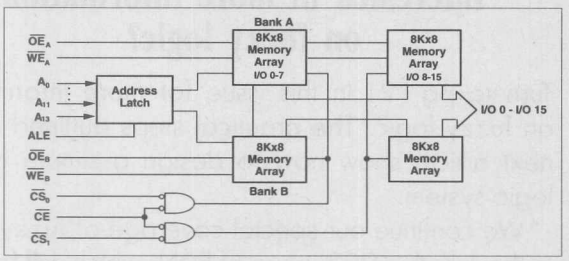
P4C188	P4C164
12ns	12ns
16Kx4	8Kx8

For Optimized Performance, Processor Specific 256K SRAMS to Support 5V and 3.3V Processors

In addition to the above fast SRAMs, Performance Semiconductor offers the P4C214, a ***processor-specific*** 16Kx16 SRAM with transparent address latches and 13ns access time for speed-critical, wide-word applications. Functionally similar to the popular 2x4Kx16 cache RAM, the P4C214 is pin-configurable for *direct-mapped* or *2-way-set* cache (supporting 386 speeds to 50MHz) and is designed for 16-bit and 32-bit bus systems, keeping chip count and power dissipation low. For use in secondary cache , the P4C214, with ALE tied high, behaves as a ***general-purpose*** 16Kx16 SRAM.

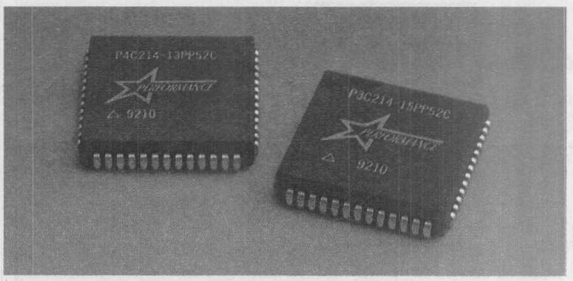


Direct Map



Two-Way Set

Very soon your Notebook PC users will demand cache . . with low power, small space, higher throughput. The 3.3V version, P3C214, with TTL-compatible I/O, uses a 3.3V power supply with access times from 12ns to 25ns. Ideal for use with a 3.3V, 16-bit bus 386SL or 386SX. Only one unit is needed for a 32KByte cache!



For more information on ultra-fast SRAMs, call Performance Semiconductor

In the United States call 408 734-9000

In Europe call 44-256-59585 (U.K.)

PERFORMANCE
SEMICONDUCTOR CORPORATION
610 East Weddell Drive, Sunnyvale, California 94089

CIRCLE NO. 69

fuzzification that available fuzzy-logic tools support.

The remaining combination/defuzzification method, the singleton technique, is a special case of the centroid method. This technique represents each fuzzy output set as a single output value by using a weighted average to combine multiple actions. This approach requires much less computation than the centroid method, but it still requires overlapping input functions to avoid discontinuities in the output. Because of its conceptual and computational simplicity—and unless someone develops a new and more powerful technique—the singleton method will probably replace the centroid method as the most common output technique.

Fuzzy feedback controllers

You can sample selected outputs of a fuzzy system and use them as inputs to make a feedback controller (**Fig 6**). A fuzzy system fuzzifies inputs, which then serve as conditions to the rule base. The rule base operates on those inputs, then it combines and defuzzifies actions from triggered rules to produce one or more controller outputs.

Increasingly, fuzzy rule-based systems use adaptive techniques—neural nets and genetic algorithms, for example—to refine degree-of-membership functions and even the actual rule bases. In addition, fuzzy decision systems that do not use a rule base are beginning to use fuzzy (rather than crisp) inputs to make predictions from vague and incomplete data.

Fuzzy logic has been applied to many different and diverse applications. These include categorization of weather patterns and of sea gull behavior; control of cement kilns, passenger trains, and elevators; scheduling of subway trains and service technicians; and making predictions in risk management.

Fuzzy logic systems are often superior to alternate approaches in five general areas:

- *In complex systems where an adequate system model is difficult or impossible to define*
- *In systems normally controlled by a human expert*
- *In systems that have moderately to very complex continuous (or semicontinuous) inputs and outputs and a nonlinear response function*
- *In systems that use human observations as control rules or inputs*
- *In systems where vagueness is common; for example, in economic systems, natural sciences, and behavioral sciences.* EDN

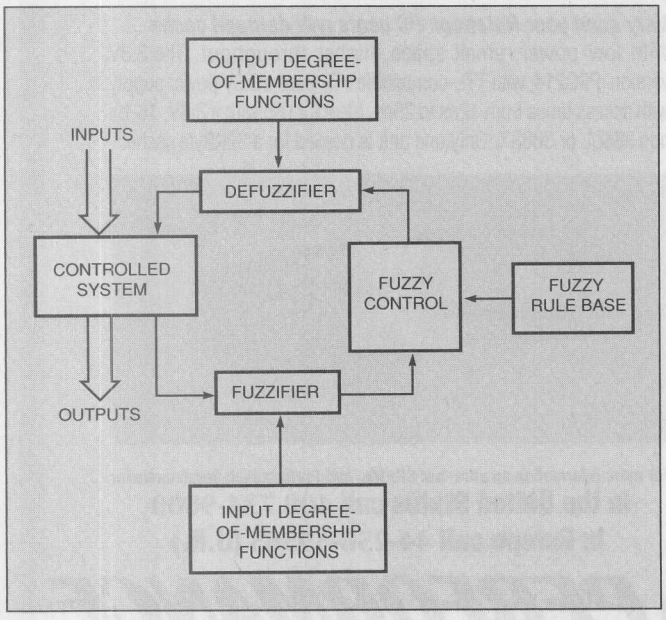
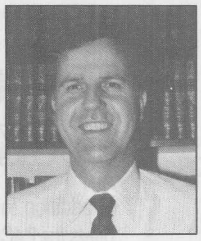


Fig 6—In a fuzzy feedback controller, selected sampled outputs serve as inputs. Modeling is unnecessary, even with nonlinear systems, as long as the response function is describable in words.

Author's biography



David Brubaker is president of The Huntington Group, consultants in the design of systems using real-time embedded processors and fuzzy logic. He has worked with Sun Microsystems, Beckman Instruments, Motorola, TRW, Ford, and ESL. David holds BS, MS, and PhD degrees, all in electrical engineering, from Stanford University. His personal activities include walking, backpacking, and coaching his children's basketball teams.

Article Interest Quotient (Circle One)
High 479 Medium 480 Low 481

Interested in more information on fuzzy logic?

Turn to pg 121 in this issue for more information on fuzzy logic. The practical steps outlined in the next article show how to design a simple fuzzy-logic system.

We continue our special coverage of fuzzy logic in the July 6, 1992, issue of EDN, which will feature a staff-written article on software tools used for fuzzy-logic design.